Simulating Application Workflows and Services Deployed on the European Grid Infrastructure

Sorina Camarasu-Pop, Tristan Glatard, Hugues Benoit-Cattin
University of Lyon, CNRS, INSERM, CREATIS, Villeurbanne, France
Email: {first.last}@creatis.insa-lyon.fr

***Abstract*—This paper presents an end-to-end SimGrid-based simulation of a Monte-Carlo computation deployed on the European Grid Infrastructure. We describe a simulation framework allowing to replay executions from real traces. Middleware services, namely the file catalog, storage elements, the DIRAC pilot-job system, and the MOTEUR workflow engine are simulated by SimGrid processes. The deployment of pilot jobs, performed on EGI by the gLite WMS and batch queues, is simulated by a random selection of platform hosts, and a matching of their latencies and failure times with real traces. The Monte-Carlo application workflow is calibrated to address performance discrepancies between the real and simulated network and CPUs. The simulation is evaluated against real executions of the GATE Monte-Carlo application. Results show that SimGrid can be used to study the performance of applications running on EGI. Simulated and real makespans are consistent, and conclusions drawn in production about the influence of application parameters such as the checkpointing frequency and the number of mergers are also made in simulation. This opens the door to better and faster experimentation on production grids.**

I. Introduction

Following the wide adoption of distributed computing systems for scientific and commercial software, applications are now deployed on platforms offering several thousands of processing units, for instance the European Grid Infrastructure (EGI[1]), BOINC platforms[2] or the Amazon Elastic Computing service[3]. Scientific experimentation with these platforms, however, remains an issue: they are subject to unpredictable load from user communities, to software and hardware failures of various types, and to dynamic resource availability. To address these difficulties, computer scientists rely on mathematical models, simulators, or experimental platforms to try to reproduce real systems in controlled conditions, which remains a challenge [1], [2].

The goal of this paper is to investigate how simulation can be used to reproduce and study executions of application workflows deployed on EGI, one of the largest distributed computing systems in the world. It is motivated by our experience with the Virtual Imaging Platform [3], a web portal supported by EGI resources, for which design choices are currently validated by unwieldy experiments on the production platform. Our ultimate goal is to be able to replay executions launched from this platform in order to understand, study and improve their performance. Our goal is not the simulation of the complete EGI system, but only of the subset of platform and services used by a particular execution of an application. In this work, we focus on the particular case of Monte-Carlo applications executed with the dynamic load-balancing method described in [4]. In [5], we studied this application on EGI to determine the impact of its checkpointing frequency, and of the number of merge jobs used to combine partial results. Here, we aim at reproducing these experiments in simulation, to investigate if similar conclusions can be drawn.

Based on the extensive, but cumbersome, experience collected on the real system, we build a simulator based on the SimGrid toolkit [6] to simulate the hardware platform, the core software services, the deployment, and the application. The platform is the set of hardware resources used by the application: storage, network, and CPU. Services are software processes that are re-used by different applications, e.g., a workflow engine, a job scheduler or a file catalog. Deployment corresponds to the matching between software services and platform entities. Finally, the application encompasses all the processes specific to a computation performed by a user.

The platform is assumed already modeled, and we focus on the services, deployment and application. Differences between the real and simulated platforms, in particular CPU and network characteristics, are compensated by calibrating the simulated application on the simulated platform.

A brief overview of related work is presented in Section II. Then, Section III describes the real system under investigation, and Section IV presents the main functionalities of SimGrid. Section V details the design of our simulation and its calibration. Section VI presents experimental results, and comparisons with the real behavior observed in [5].

II. Related work

There exists a variety of simulation toolkits, among which OptorSim [7], GridSim [8], PeerSim [9], CloudSim [10], and SimGrid [6]. Recent reviews of these tools are available, e.g., in [11] and in [12] which focuses on peer-to-peer platforms. The related work presented below is structured as Sections III, IV and V, covering literature on (i) the platforms used in simulations, (ii) simulated grid services, (iii) application deployment and (iv) types of applications that have been simulated recently.

Platforms used in simulators are usually synthetized or modeled from existing ones [13], [14], for instance using network tomography [15] to build a network model from a

[1]http://www.egi.eu
[2]http://boinc.berkeley.edu
[3]http://aws.amazon.com/ec2

real platform. OptorSim has a model of the EU Datagrid platform (an ancestor of EGI), but it focuses on inter-site communications [16]; computing sites are modeled as single processing units that can only process one job at a time [17].

Various kinds of grid services were simulated. For instance, [18] reports on a simulation involving a DIRAC pilot-job scheduler, and OptorSim and the work in [19] simulate data management services. A few MapReduce tools have been simulated too; for instance, [20] simulates a Hadoop cluster and uses it to study the impact of data locality, network topology, and failures on applications. In [21], the authors propose a MapReduce simulator using GridSim.

The simulation of applications deployment, in particular scheduling, has received a lot of attention and custom tools have been developed, including GridSim [22], and SimGrid [23] which was initially designed for the simulation of schedulers [24].

Different types of application models have been simulated. SMPI [25] is a SimGrid-based simulator for MPI applications in which the application is actually executed, but part of the execution is wrapped by the simulator. The work in [26] simulates bag-of-task applications to study their scalability. In [27], the authors use simulation to study the checkpointing of sequential applications to improve fault-tolerance.

To the best of our knowledge, no simulation of applications using computing and storage resources on the production system of the European Grid Infrastructure has been performed. In the following sections, we describe our attempt to do so using SimGrid. It is an active software project on which consistent efforts have been devoted to scalability, and which exposes a rich simulation toolkit through well-documented, easy and complete APIs.

III. Real System to Simulate

We consider a system made of a complete Monte-Carlo application deployed on the European Grid Infrastructure using the MOTEUR workflow engine and the DIRAC job scheduler.

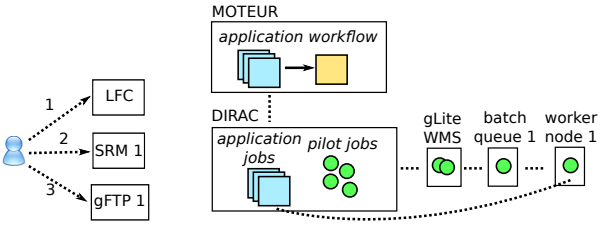
A. Platform

The target platform is the biomed Virtual Organization (VO) of EGI. This VO is supported by about 100 computing sites world-wide, offering a total of 179 computing clusters, and storage. Sites are interconnected by non-dedicated academic networks, typically NRENS and GEANT[4] in Europe. A detailed description of the biomed VO infrastructure is provided by EGI[5]; 4 PB of storage is available, and the average number of simultaneously running jobs during the last year was 1320.

On this platform, application jobs are often delayed by a few minutes to a few hours before being executed. We call this delay *latency*, which encompasses scheduling time, queuing time in batch systems and other overheads. Application jobs are also subject to failures that can occur at any time due to software or hardware issues.

[4]http://www.geant.net

[5]http://gstat.egi.eu/gstat/gstat/vo/biomed



(a) File transfer (b) Workload management.

Fig. 1. Services and deployment on the real system

B. Services

As summarized in Figure 1, the platform is accessed through a set of core services offering basic operations for job and storage management. Figure 1(a) illustrates file transfer services and their interactions. Storage resources are accessed through a three-layer stack. At the lowest level, files are transferred using the gridFTP transfer protocol. On top, the Storage Resource Manager (SRM) schedules file transfers on the gridFTP backends available on the site. Finally, a central logical file catalog (LFC) stores information about file replicas, and provides a single logical addressing space for distributed storage. File transfers are performed by gLite clients wrapping operations involving these 3 entities [28]. Numbers on Figure 1(a) indicate steps of a file download: (1) storage site is obtained from logical file name by querying the LFC; (2) transfer request is issued to SRM; (3) transfer is done through a gridFTP server. File upload follows a similar reverse sequence.

Figure 1(b) gives an overview of workload management services and their deployment. Jobs are scheduled on computing resources using the DIRAC scheduler [29]. This service provides a pilot-job framework where application jobs are not directly submitted to the computing sites, but assigned to pilot jobs that are already running on resources provisioned separately. Latency and failures can therefore be reduced.

Application jobs are generated, submitted and monitored by the MOTEUR workflow manager [30] from a description of the application workflow. MOTEUR jobs transfer the input data, perform various checks on the computing resource, run the application code itself, and finally upload the results on the site storage elements. MOTEUR also measures parameters about the execution, such as the latency and CPU power of each job, and timestamps of various execution phases (see details in [31]). Failed jobs are automatically resubmitted up to a configurable number of times.

C. Deployment

Deployment of storage services is done statically by the resources providers. Typically, each site hosts a storage element made of one SRM server backed by one or several gridFTP servers. A single central LFC is available in the biomed VO.

The MOTEUR workflow engine and DIRAC scheduler are also statically deployed on hosts of the same network. Pilot jobs, however, are dynamically deployed through the resource-provisioning mechanism implemented in DIRAC: they are

submitted either directly to queues of the sites' batch schedulers or through the gLite Workload Management System (WMS) which dispatch them on sites. Dirac also plays a role in this dispatching, by using site blacklists defined by platform administrators. The pilot deployment process results from a combination of meta and local scheduling processes configured by various admins, which can therefore hardly be described as a single algorithm.

D. Application

The considered application is the GATE Monte-Carlo code for medical simulation [32]. It is implemented as a MOTEUR workflow available on the VIP/Gate-Lab[6] web platform. Monte-Carlo applications consist in the computation of several independent operations called events. The workflow mainly consists of (i) a computing part, when partial simulation results are produced by Monte-Carlo jobs, and (ii) a merging part, where these results are assembled together. We call Monte-Carlo phase the phase of the execution where there is at least one active Monte-Carlo job, and merging phase the phase where only merge jobs are active.

The computing phase is implemented using the dynamic load-balancing method described in [4], where each Monte-Carlo job contributes to the simulation uninterruptedly until it is stopped by the workflow engine. In these conditions, each computing resource executes at most one Monte-Carlo job. All jobs are single-core. Two execution modes are considered: with checkpointing and without checkpointing.

Without checkpointing, merge jobs are launched once all the Monte-Carlo jobs are finished, while with checkpointing, they are launched from the beginning of the execution. In the latter case, merge jobs are active during the Monte-Carlo phase. To parallelize data transfers, several merge jobs can run concurrently. They rely on the central LFC to handle concurrency issues using logical directory creation/deletion as locking mechanism. The two application workflows, with and without checkpointing, are described in Figures 2 and 3 that are adapted from [5].

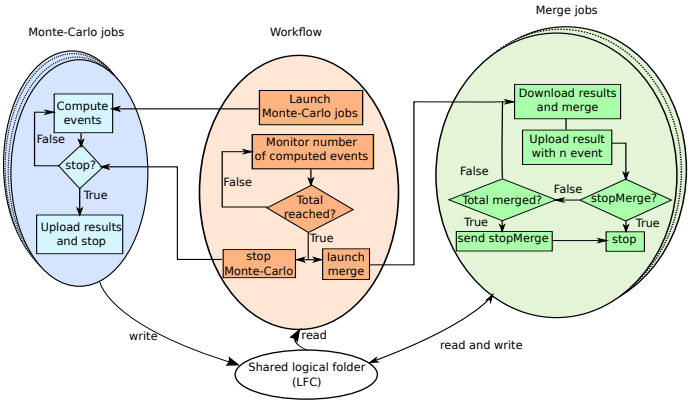


Fig. 2. Monte-Carlo workflow without checkpointing (adapted from [5]). Monte-Carlo jobs upload their results once at the end of their lifetime. The stop Monte-Carlo condition triggers: i) results upload from Monte-Carlo jobs and ii) the launch of merging jobs. The merge stop signal is sent by the first merging job having merged the required number of events.

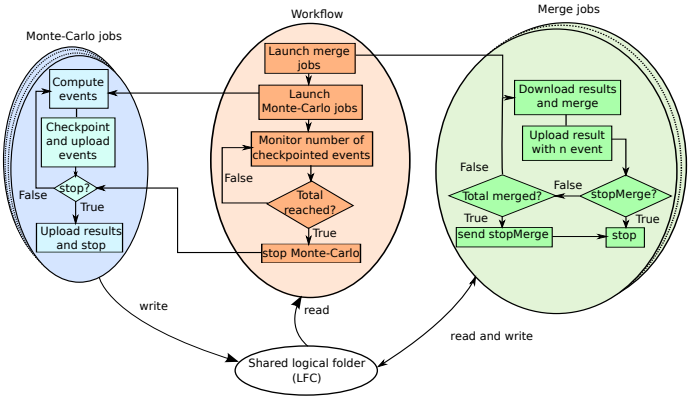


Fig. 3. Monte-Carlo workflow with checkpointing (adapted from [5]). Multiple Monte-Carlo and merge jobs are launched in parallel at the beginning of the workflow. Monte-Carlo jobs upload their results regularly, at the same frequency as they checkpoint their partial results. The stop Monte-Carlo condition given by the master triggers the end of Monte-Carlo jobs. From this moment on, only merging jobs continue to run (merging phase). The stopMerge signal is sent by the first merging job having merged the required number of events.

IV. The SimGrid toolkit

SimGrid [6], [33] is a toolkit that provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments. This section describes the functionalities used in our simulation.

Platform: a platform in SimGrid consists of a hierarchical description of Autonomous Systems (AS). Each AS can contain (i) hosts, (ii) routers, (iii) links defining the connection between two (or more) resources (links have a bandwidth and a latency) and (iv) clusters, which can contain a certain number of hosts interconnected by some dedicated network. The routing between the elements of an AS has to be explicitly defined. Network is simulated using a fluid network model [33]. Hosts define physical resources with computing capabilities, one or more mailboxes to enable communication, and some private data that can be only accessed by local processes. A host is identified by its id and peak power expressed in flop/s. An availability value can also be defined for the host along time; it expresses the percentage of computing power available. The ON/OFF state of the host can also be described along time. More information about platform description in SimGrid is available in [16].

Services: task transfer and execution are provided by SimGrid APIs. Tasks are defined by a computing amount, a message size and some private data. Tasks can account for both processing and data transfer (potentially with null computing amount). Task exchanges are handled like messages that can be sent and received from/to mailboxes. Once received, tasks are handled by *processes*, that may compute them or transfer them to others hosts. SimGrid offers several interfaces, among which MSG (Meta-SimGrid) for multi-agents simulations, SMPI for MPI simulations, and SimDag for DAGs of parallel

[6]https://vip.creatis.insa-lyon.fr

tasks.

Deployment: deployment defines where the simulator deploys the processes. It can be programmed using specific functions, or defined in a XML file which associates processes to hosts and specifies their arguments, for instance the list of workers of a master, or the site storage element.

V. Simulation Design

Table I compares the real and simulated systems. The simulation design is detailed below.

A. Platform

Despite various attempts to model the EGI platform, there is currently no suitable model available for our simulations. This is mainly due to the lack of detailed information on (i) the configuration (number and performance of processing units, intra-site bandwidth, etc.) of its more than 300 distributed sites and (ii) the inter-site network connections. Gathering the necessary information is particularly challenging because it changes frequently.

Consequently, we used the Grid5000 platform[7] model. It comprises 10 sites (AS), 40 clusters, approximately 1500 nodes, and the network infrastructure of the Grid5000 platform. We added a single-host cluster to each site of the platform to deploy its storage element; the network link used to connect this cluster had similar performance to the other links in the site.

B. Services

Two SimGrid processes were implemented to simulate (i) the LFC and (ii) an SE composed of an SRM and gridFTP service. They simulate communications involved in file transfer operations using the SimGrid MSG API. The main messages handled by our simulated LFC are file registration, directory management, and file replica listing. Only a single replica per file can be handled at the moment. Our SE has two operations: file upload and file download. File transfers are implemented as tasks with null computing amount sent from the transfer source to the destination. I/O operations on the storage disks are assumed negligible with respect to cross-site transfers. Two methods were defined to simulate the gLite file transfer clients: (i) copy and register file, which consists of file upload to SE and registration in LFC, and (ii) copy file, which consists of file replica listing in LFC and file download from SE.

A master and a worker process were implemented to simulate job creation, scheduling and workload execution. The master simulates both the MOTEUR workflow engine and the DIRAC scheduler. It first initializes the workers by sending them an `init` task with null computing amount and null data transfer. It then generates the Monte-Carlo jobs and sends them to worker processes as soon as they acknowledge the `init` task (first come, first served). If checkpointing is enabled, the master sends merge jobs to the workers from the beginning of the simulation; otherwise, it waits for the Monte-Carlo jobs to be finished.

[7]https://www.grid5000.fr

Worker processes simulate both DIRAC pilot jobs and the application jobs. They declare themselves to the master by sending an `ACK` message in response to `init`. They can process both Monte-Carlo and merge jobs. All processes are implemented using SimGrid's MSG API.

C. Deployment

The resource provisioning process, implemented by the gLite WMS, site batch queues and DIRAC pilot submitter in the real system, is simulated by randomly generating the list of hosts used to deploy workers for each simulation. Two separate host lists are used, one for each type of application worker. The master and LFC processes are deployed once and for all on random nodes in the platform.

Job latencies and failures are simulated using SimGrid's host availability and state files. This is possible because each platform host executes at most one Monte-Carlo job (see Section III). A job latency of t is modeled by a host availability of 0 until time t. A job failure at time t is modeled by a host going from state ON to OFF at time t. Latency values and failure times are extracted from real traces. To match latency/failure times extracted from real jobs to availability/state values of simulated hosts, we do a pairwise match between the list of real jobs sorted by decreasing values of measured CPU power and the list of simulated hosts sorted by decreasing values of assigned CPU power. This matching is done separately for each type of application worker. Job resubmissions triggered by MOTEUR when jobs fail are simulated as any other job.

D. Application

Two types of application workers are simulated, one for Monte-Carlo jobs and one for merge jobs. Monte-Carlo jobs download their input data using the file transfer method, and start the Monte-Carlo simulation. They periodically report their number of computed events to the master by sending `events` messages. If checkpointing is enabled, they also upload their results to the site's SE using the file transfer method. Monte-Carlo jobs terminate and upload their final result when they receive a `stop` message from the master. A fixed file size is used for all Monte-Carlo results.

To handle concurrency among mergers, an atomic operation is implemented in the LFC process to return a unique list of files to merge at each invocation. Mergers invoke this operation, and download the returned files. The merging process itself is simulated by sleeping during the CPU time of the merging operation. The merged result is finally uploaded using the file transfer method. In case it contains more than the total number of events to compute, a `stopMerge` directory is written in the LFC. Merge processes periodically check the presence of this directory and terminate accordingly.

The master receives `events` messages specifying the number of Monte-Carlo events computed by the emitter. Based on these, it maintains the total number of computed events, and sends a `stop` message to all Monte-Carlo processes when all the events are computed.

Function		Real system	Simulated system
Application description		MOTEUR workflow with parameters: • checkpointing period • number of mergers • number of Monte-Carlo jobs • number of Monte-Carlo events	Master and worker processes with parameters: • checkpointing period • number of mergers • number of Monte-Carlo jobs • number of Monte-Carlo events • computational cost of Monte-Carlo event * • computational cost of merge operation * • file size *
Services	File catalog File transfer Job scheduling Workload	LFC SRM and gridFTP MOTEUR and DIRAC Pilot and application jobs	LFC process SE process Master process Worker process
Deployment		gLite WMS and batch queues	Random selection of hosts for worker processes Latencies and failure times matched to real trace
Platform		EGI, biomed VO, 100 sites, 179 clusters, non-dedicated network	Grid'5000 platform model 10 sites, 40 clusters, dedicated network

TABLE I
OVERVIEW OF REAL AND SIMULATED SYSTEMS. * CALIBRATED PARAMETERS

E. Calibration

Calibration of application parameters is required to address performance discrepancies between the real and simulated platforms. We use the following rule to determine the computing cost of a Monte-Carlo event:

$$x[flop/event] = \frac{\bar{p}_{sim}[flop/s]CPU_{real}[s]}{E_{real}[event]}, \quad (1)$$

where the units are given in square brackets, x is the computational cost of an event, $\bar{p}_{sim}$ is the average performance of hosts on the simulation platform, CPU_{real} is the cumulative CPU time of the Monte-Carlo computation on the real platform, and E_{real} is the number of Monte-Carlo events in the real application. We consider real executions of a GATE Monte-Carlo computation of 50 million events which represents 60,000 minutes of CPU on the biomed VO. The Grid'5000 platform model used here has an average performance $\bar{p}$ of 14 Gflop/s, which for our calibrating application gives $x = 1$ Gflop/event.

We use a similar rule to determine the computing cost of a merge operation between two files:

$$y[flop/merge] = \bar{p}_{sim}[flop/s]CPUm_{real}[s], \quad (2)$$

where $CPUm_{real}$ is the CPU time of a merge operation on the real platform. We obtain $y = 80$ Gflop for our application and simulated platform.

The size of the files produced by Monte-Carlo jobs was calibrated so that the simulated file transfer times in merge jobs match best the median time measured on the real application. Figure 4 compares file transfer times in the merge jobs for the real and simulated experiments. Three file sizes, 10MB, 15MB and 20MB were tested. Based on this, we chose a file size of 15MB, which has a median transfer time very close to the real one for download (34s vs 35s) and reasonably close for upload (26s VS 18s).

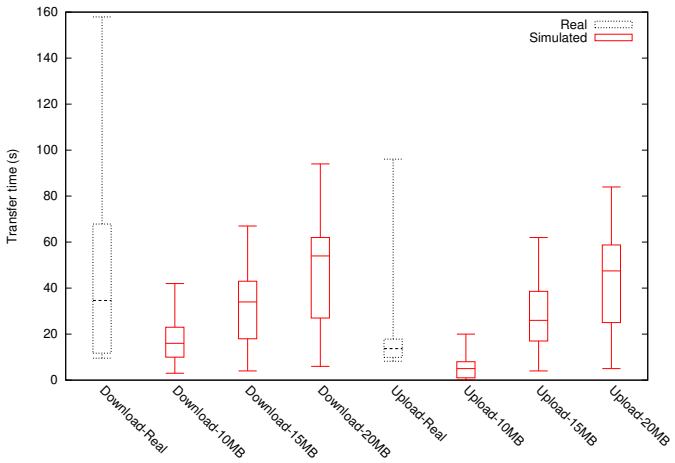


Fig. 4. Calibration of the file size. A file size of 15MB was chosen.

VI. EXPERIMENTS AND RESULTS

We made some experiments to compare the real and simulated execution time for different numbers of mergers and different checkpointing periods. For each experiment, the Monte-Carlo phase was executed by 300 parallel jobs. Table II summarizes the tested application configurations. Three real traces are available for each configuration, and 5 simulations were performed for each real trace. For a given real trace, simulations only differ by the random list of hosts used to deploy the application workers.

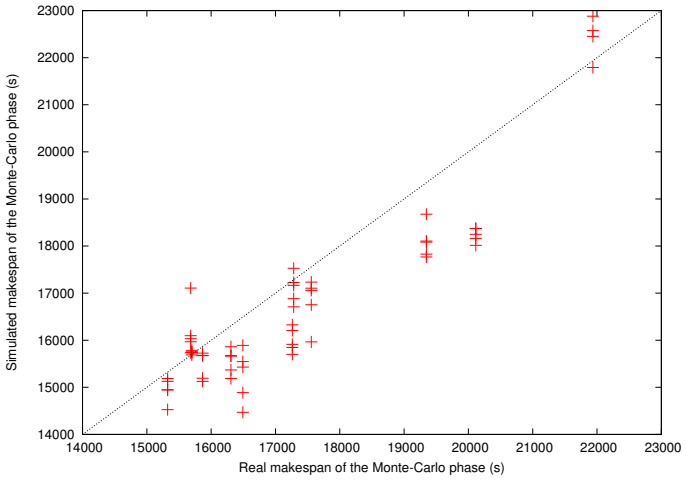


Fig. 5. Real versus simulated makespans of the Monte-Carlo phase for the workflow without checkpointing, and the 4 merger configurations.

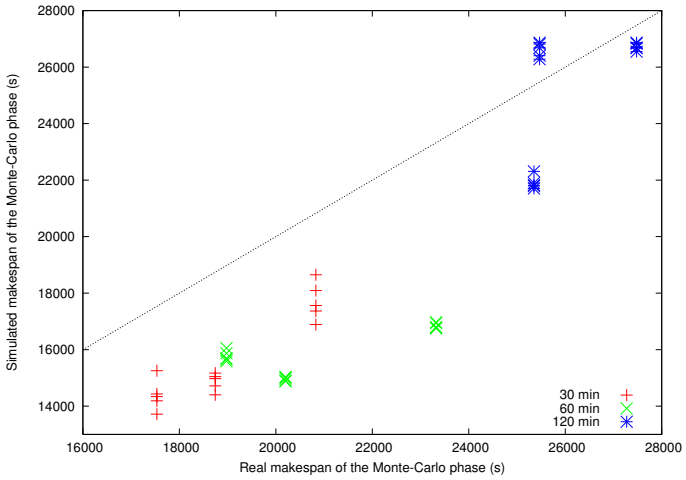


Fig. 6. Real versus simulated makespans of the Monte-Carlo phase for the workflow with checkpointing (10 mergers).

A. Study of the Monte-Carlo phase

1) No checkpointing: Figure 5 shows an XY plot of the makespan of the Monte-Carlo phase in the simulated and real systems for the workflow without checkpointing. Results show a good match between the values of the simulation and the real experiments; the root mean-square error (RMSE) is 1016s, and the average relative error is 4.4%. Such a good match is obtained from the real latency and failure time values injected in the simulation, and the calibration of the application.

2) With checkpointing: Figure 6 shows an XY plot of the makespan of the Monte-Carlo phase in the simulated and real systems for the workflow with checkpointing. The match is still good, but the error is larger; RMSE is 3785s, and the average relative error is 16%. The error is more important for executions with short checkpointing periods (30 and 60 minutes), where the transfer time of partial results is significant. For the workflows with a checkpointing period of 120 minutes, simulation is more accurate, with an average relative error of 7%. This suggests that a significant part of the error comes from the simulation of data transfers, which is expected given that the platform model used in the simulations has a different network topology than the real one.

Figure 7 shows real and simulated computing times as a function of the checkpointing period. We notice that the simulation and real values follow the same trend: the makespan increases with the checkpointing period, which is consistent with the observations in [5].

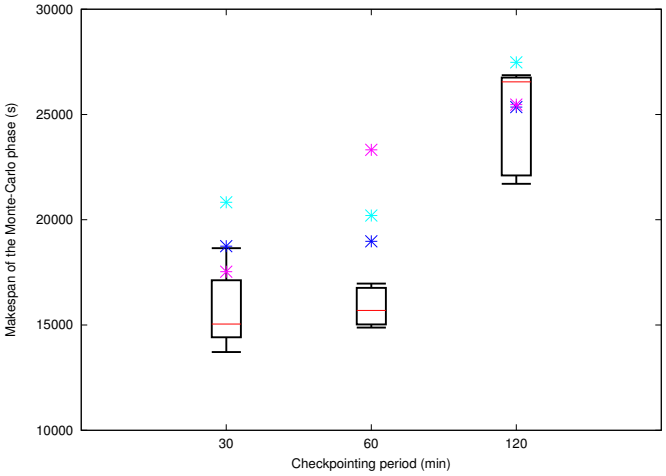


Fig. 7. Real and simulated makespans of the Monte-Carlo phase as a function of the checkpointing period for the workflow with checkpointing. The 9 real values are shown individually, while the 45 simulated values are grouped in three box-and-whisker plots with median values, one for each checkpointing frequency.

# mergers	checkpointing period
1	disabled
5	disabled
10	disabled
15	disabled
10	30min
10	60min
10	120min

TABLE II
APPLICATION CONFIGURATIONS TESTED IN THE EXPERIMENTS.

B. Study of the merging phase

Figure 8 shows the real and simulated makespans of the merging phase as a function of the number of mergers for the workflow without checkpointing. The simulated and real makespans follow the same trend: merging time decreases when the number of parallel mergers increases, until the threshold of 10 mergers after which it slightly increases.

The simulated makespan is less variable than the real one for one and five mergers. This is most likely because the Grid5000 platform used in the simulation is significantly less heterogeneous that the EGI used for the real executions. The performance variability coming from the random choice of

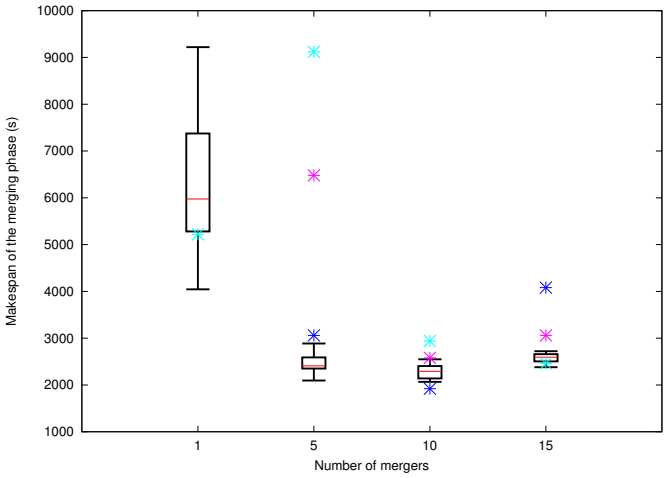


Fig. 8. Real and simulated makespans of the merging phase as a function of the number of mergers for the workflow without checkpointing. The 12 real values are shown individually, while the 60 simulated values are regrouped in four box-and-whisker plots with median values, one for each number of mergers. Due to scale issues, only 1 real value is displayed for 1 merger; the two others are at 19980 s and 26760 s.

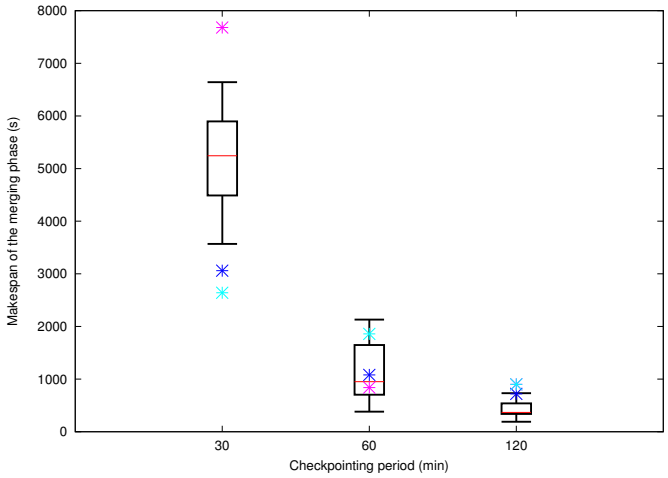


Fig. 9. Real and simulated makespans of the merging phase as a function of the checkpointing period for the workflow with checkpointing. Real values are shown individually, while the 45 simulation values are grouped in three box-and-whisker plots with median values, one for each checkpointing period.

hosts in the simulated deployment remains lower than the variability encountered in reality.

When there is only one merger, the merging time is tightly dependent on the performance of the selected node. This explains the higher variability of the makespan for one merger. When the number of mergers increases (10 or 15), the merging performance is less sensitive to heterogeneity since a few good hosts will complete the work not accomplished by the slow ones. Results for the real executions run with the same parameters are thus closer. In these cases the simulation matches very well the real observations.

Figure 9 shows the real and simulated makespans of the merging phase as a function of the checkpointing period for the workflow with checkpointing. Simulation results are close to real values and follow the same trend: merging time decreases when the checkpointing period increases because the number of partial results to merge decreases. For a checkpointing period of 30 minutes, the performance of the real application is very variable. This is a borderline case (as was the workflow without checkpointing and one merger) for which platform heterogeneity plays an important role. The simulation manages to capture these differences only to a certain extent, simulation values showing more variability than for higher checkpointing periods.

VII. Conclusion

We presented a simulation framework for a Monte-Carlo application workflow deployed on EGI with the MOTEUR engine and DIRAC pilot jobs accessing data on storage elements via the LFC file catalog. The considered application and deployment are complex and similar to many deployments in EGI. Based on the SimGrid toolkit, we simulate the deployment of pilot jobs by randomly selecting hosts and matching latencies and failure times to real traces. Job scheduling, application jobs and the main EGI data management services (file catalog and storage element) are also simulated as SimGrid processes. To address performance discrepancies between the real and simulated platforms, the computational cost and file size of the application are calibrated from real traces.

The resulting simulator is used to study the influence of the number of mergers in the application workflow, and its checkpointing period. Results show that absolute makespan values of the Monte-Carlo phase are consistent, and that the trends observed in production are correctly reproduced by the simulation: (1) increasing the checkpointing period increases the makespan of the Monte-Carlo phase, and it decreases the makespan of the merge phase, and (2) increasing the number of mergers decreases the makespan of the merging phase up to a threshold beyond which it increases again. In addition, the simulation reveals that the performance of the merging phase is highly impacted by the heterogeneity of the platform, which could hardly be observed in production.

We therefore conclude that SimGrid can be used to simulate performance studies of applications running on EGI. In view of the technical expertise, time and human cost required to perform meaningful performance studies in production conditions, this opens the door to much better and faster evaluations for applications deployed on this platform.

The good match between real and simulated performance is explained by the realism of SimGrid, by our calibration of the application on the simulated platform, and by the injection of real latency and failure values in the simulation. Overall, our results show that realistic simulated behaviors may be obtained on a simulated platform that corresponds to the real one only to a limited extent. We believe that this is an interesting conclusion given the difficulty to build realistic models of real large-scale platforms. Nevertheless, further investigations are

needed to determine to what extent (in which conditions) a simulated platform can be used as a model of a real platform.

The objective of replaying in simulation a Monte-Carlo computation executed with dynamic load-balancing on EGI using traces extracted from the Virtual Imaging Platform is fulfilled. However, much remains to be done to replay *any* workflow execution. For Monte-Carlo applications, a static load balancing scheme, where tasks compute a fixed number of events, should also be implemented since it corresponds to the most used configuration (although sub-optimal). In addition, a more elaborated workflow engine should be simulated to enable the simulation of any application available in the VIP/Gate-Lab web platform.

Acknowledgements

We warmly thank the SimGrid development and support team[8] for their help and support with the simulator. We also thank the site administrators of the European Grid Initiative and the GGUS support for their help. This work falls into the scope of the scientific topics of the French National Grid Institute (IdG).

References

[1] Gustedt, Jeannot, and Quinson, "Experimental validation in large-scale systems: a survey of methodologies," *Parallel Processing Letters*, vol. 19, no. 3, pp. 399–418, 2009.

[2] Epema, "Twenty years of grid scheduling research and beyond (keynote talk)," in *CCGrid'2011*, Ottawa, CA, may 2012, pp. xxxi – xxxiii.

[3] Glatard, *et al.*, "A virtual imaging platform for multi-modality medical image simulation," *IEEE Transactions on Medical Imaging*, vol. in press, 2012.

[4] Camarasu-Pop, Glatard, Moscicki, Benoit-Cattin, and Sarrut, "Dynamic partitioning of GATE Monte-Carlo simulations on EGEE," *Journal of Grid Computing*, vol. 8, no. 2, pp. 241–259, mar 2010.

[5] Camarasu-Pop, Glatard, Silva, Gueth, Sarrut, and Benoit-Cattin, "Monte-carlo simulation on heterogeneous distributed systems: a computing framework with parallel merging and checkpointing strategies," *Future Generation Computer Systems*, 2012, in press.

[6] Casanova, Legrand, and Quinson, "SimGrid: a Generic Framework for Large-Scale Distributed Experiments," in *10th IEEE International Conference on Computer Modeling and Simulation*, Mar. 2008.

[7] Bell, Cameron, Capozza, Millar, Stockinger, and Zini, "Optorsim - a grid simulator for studying dynamic data replication strategies," *International Journal of High Performance Computing Applications*, 2003.

[8] Buyya and Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience (CCPE)*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[9] Montresor and Jelasity, "Peersim: A scalable p2p simulator," in *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, sept. 2009, pp. 99 –100.

[10] Calheiros, Ranjan, Beloglazov, De Rose, and Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[11] Beaumont, *et al.*, "Towards Scalable, Accurate, and Usable Simulations of Distributed Applications and Systems," INRIA, Rapport de recherche RR-7761, Oct. 2011.

[12] Naicken, Basu, Livingston, and Rodhetbhai, "A survey of peer-to-peer network simulators," in *Proceedings of The Seventh Annual Postgraduate Symposium*, 2006.

[13] Lu and Dinda, "Synthesizing realistic computational grids," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, ser. SC '03. New York, NY, USA: ACM, 2003, pp. 16–.

[14] Quinson, Bobelin, and Suter, "Synthesizing generic experimental environments for simulation," *International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing*, pp. 222–229, 2010.

[15] Castro, Coates, Liang, Nowak, and Yu, "Network tomography: Recent developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, aug 2004.

[16] Frincu, Quinson, and Suter, "Handling Very Large Platforms with the New SimGrid Platform Description Formalism," INRIA, Rapport Technique RT-0348, 2008.

[17] Mairi and Nicholson, "File management for hep data grids," Tech. Rep., 2006.

[18] Caron, Garonne, and Tsaregorodtsev, "Definition, modelling and simulation of a grid computing scheduling system for high throughput computing," *Future Gener. Comput. Syst.*, vol. 23, no. 8, pp. 968–976, Nov. 2007.

[19] Sulistio, Cibej, Venugopal, Robic, and Buyya, "A toolkit for modelling and simulating data grids: an extension to gridsim," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 13, pp. 1591–1609, 2008.

[20] Wang, Butt, Pandey, and Gupta, "A simulation approach to evaluating design decisions in mapreduce setups," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, sept. 2009, pp. 1 –11.

[21] Teng, Yu, and Magoule ands, "Simmapreduce: A simulator for modeling mapreduce framework," in *Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on*, june 2011, pp. 277 –282.

[22] Buyya, Murshed, Abramson, and Venugopal, "Scheduling parameter sweep applications on global grids: a deadline and budget constrained costtime optimization algorithm," *Software: Practice and Experience*, vol. 35, no. 5, pp. 491–512, 2005.

[23] Santos-Neto, Cirne, Brasileiro, and Lima, "Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids," in *Job Scheduling Strategies for Parallel Processing*, 2005, vol. 3277, pp. 210–232.

[24] Legrand, Marchal, and Casanova, "Scheduling distributed applications: the simgrid simulation framework," in *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, may 2003, pp. 138 – 145.

[25] Clauss, Stillwell, Genaud, Suter, Casanova, and Quinson, "Single node on-line simulation of mpi applications with smpi," in *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, may 2011, pp. 664 –675.

[26] Silva and Senger, "Scalability limits of bag-of-tasks applications running on hierarchical platforms," *Journal of Parallel and Distributed Computing*, vol. 71, no. 6, pp. 788 – 801, 2011.

[27] Bouguerra, Kondo, and Trystram, "On the scheduling of checkpoints in desktop grids," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, may 2011, pp. 305 –313.

[28] Laure, *et al.*, "Programming the Grid with gLite," *Computational Methods in Science and Technology*, vol. 12, no. 1, pp. 33–45, 2006.

[29] Tsaregorodtsev, *et al.*, "DIRAC3 . The New Generation of the LHCb Grid Software," *Journal of Physics: Conference Series*, vol. 219 062029, no. 6, 2009.

[30] Glatard, Montagnat, Lingrand, and Pennec, "Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR," *International Journal of High Performance Computing Applications (IJHPCA)*, vol. 22, no. 3, pp. 347–360, Aug. 2008.

[31] Silva and Glatard, "A Science-Gateway Workload Archive to Study Pilot Jobs, User Activity, Bag of Tasks, Task Sub-Steps, and Workflow Executions," in *CoreGRID/ERCIM Workshop on Grids, Clouds and P2P Computing*, Rhodes, GR, 2012.

[32] Jan, *et al.*, "Gate v6: a major enhancement of the gate simulation platform enabling modelling of ct and radiotherapy," *Physics in medicine and biology*, vol. 56, no. 4, pp. 881–901, 2011.

[33] Bobelin, Legrand, David, Alejandro González, Navarro, Quinson, Suter, and Thiery, "Scalable Multi-Purpose Network Representation for Large Scale Distributed System Simulation," in *CCGrid 2012 – The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Ottawa, Canada, May 2012, p. 19, rR-7829 RR-7829.

[8] http://simgrid.gforge.inria.fr